GotABot

User Guide version 4.6.0

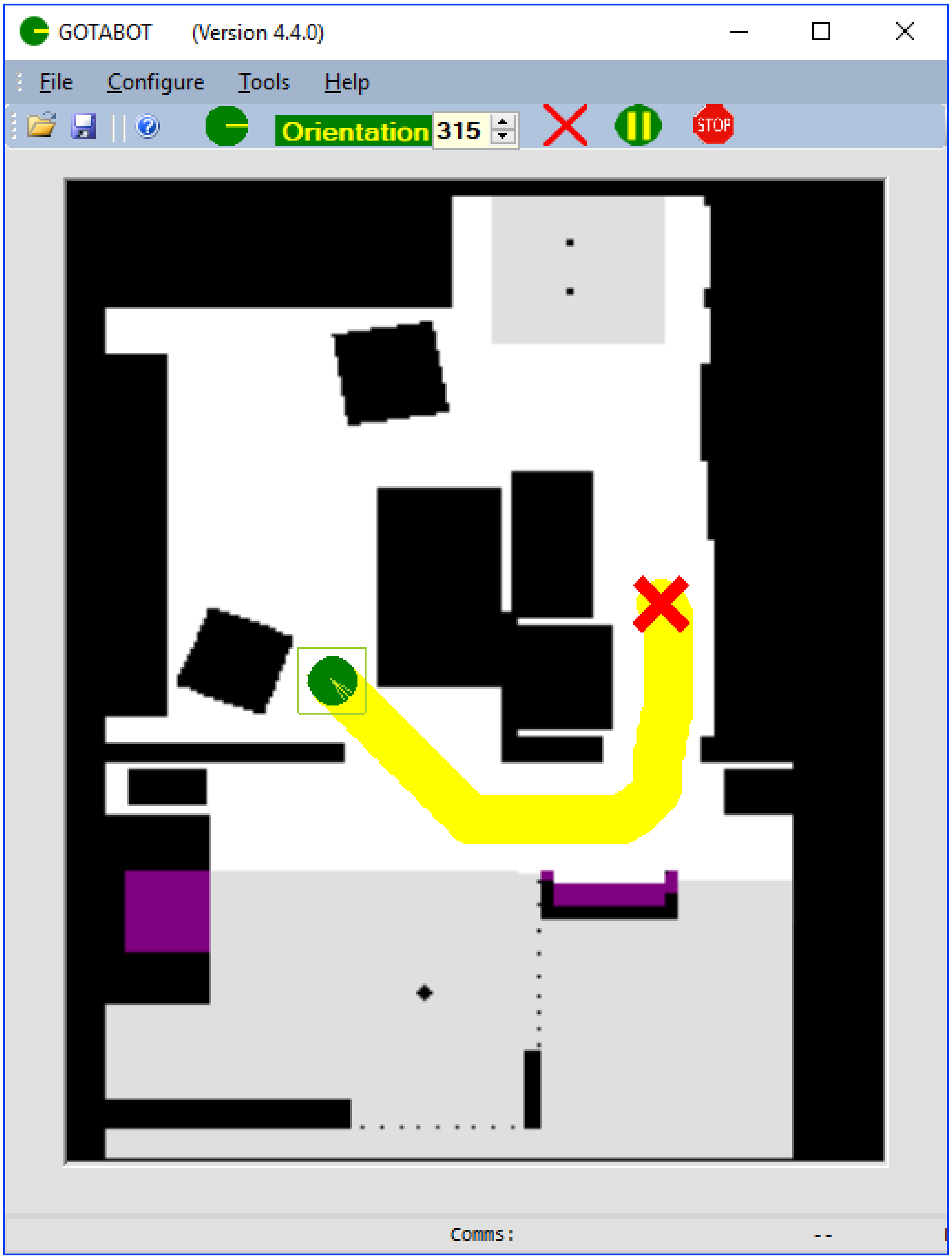


Table of Contents

1. Introduction

So you've got a robot and need some software to make it do stuff. Perhaps GotABot can help with that. Using a map, it can plan a route and direct your robot to a desired destination. With odometry and a camera , Kinect or laser scanner, GotABot will track the robot and correct its position and trajectory so as to stay on course.

GotABot also supports optional enhancements such as an arm, bumper switches, Kinect or URG04LX Laser Range Finder.

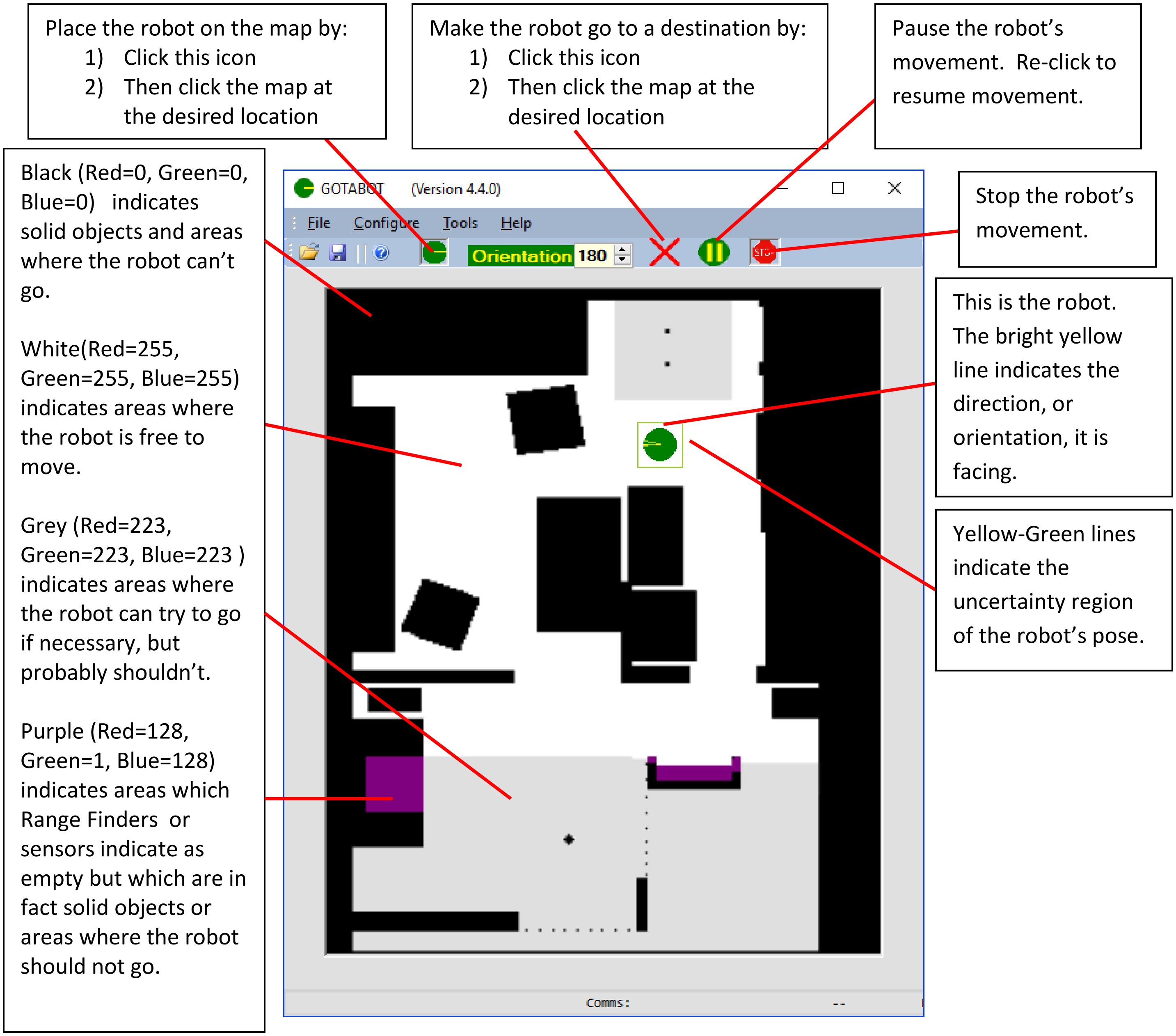
Additional information and the latest version of the software may be found on the website: http://gotabot.weebly.com/.

Important Safety Information

Although great effort has been taken to debug GotABot, it is a "Use at your own risk" type program. Please use common sense and follow all safety instructions provided by the robot manufacturer.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE

2. Getting Started



Place the robot on the map by:
1) Click this icon
2) Then click the map at the desired location

Make the robot go to a destination by:
1) Click this icon
2) Then click the map at the desired location

Pause the robot's movement. Re-click to resume movement.

Stop the robot's movement.

Black (Red=0, Green=0, Blue=0) indicates solid objects and areas where the robot can't go.

White(Red=255, Green=255, Blue=255) indicates areas where the robot is free to move.

Grey (Red=223, Green=223, Blue=223) indicates areas where the robot can try to go if necessary, but probably shouldn't.

Purple (Red=128, Green=1, Blue=128) indicates areas which Range Finders or sensors indicate as empty but which are in fact solid objects or areas where the robot should not go.

This is the robot. The bright yellow line indicates the direction, or orientation, it is facing.

Yellow-Green lines indicate the uncertainty region of the robot's pose.

Figure 1 Overview

QuickStart

Choose Your Robot

If Using an Evolution Robotics ER1 robot

1) Configure GotaBot and the ER1
2) Choose the localization method and configure

If Using a Heathkit Hero-1 robot

1) Use the Heathkit Hero-1 setup guide

Create a map

1) Create a map

Choose Your Sensor

If Using a Kinect or URG04LX Laser Range Scanner or a Heathkit Hero 1 robot

1) Configure the Scanner.
2) Configure various variables
3) Operate

If Using a Webcam and LandMarks

1) Setup some Landmarks
2) Configure various variables
3) Operate

Then What?

Once basic operation is achieved the user may explore this manual for additional capabilities and options.

TIP: Easy Localization

Localization using a Laser Scanner or Kinect is easier, faster and more accurate than localization using a camera and landmarks.

System Configuration and Installation of GotABot

System Configuration

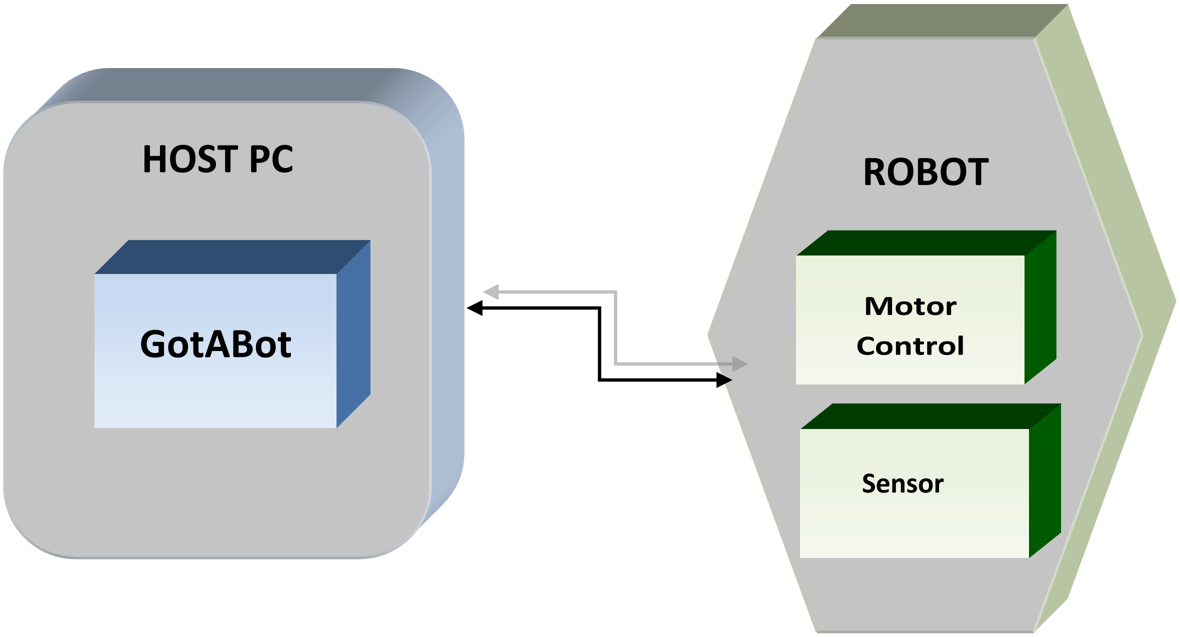
Before installing GotABot, one should first decide how to configure one's robotic system. There are a number of factors to consider such as:

1) The robot which is to be controlled
2) Will GotABot act as a standalone controller or will it be controlled by a higher level program.
3) Will the robot use a Kinect or laser scanner or a webcam to determine its position.

These and other considerations will be discussed in subsequent chapters.

Simple System Configuration

The simplest system configuration is as shown below. The Host PC may be separate from the robot and communicate to it via a wireless link or it might be on or part of the robot itself. This is the easiest configuration and should,be used first if for no other reason than to familiarize oneself with GotABot operation.



Other System Configurations

Although the previous ‘Simple System Configuration’ works well, there are other configurations which may be more optimal for some applications:

1) If you wish to add additional behaviors, customize GotABot’s operation, or allow a separate program to use all or portions of GotABot’s functionality then see the chapter “API“.
2) A Kinect or URG04 Scanner is easier to setup and can provide better performance than image recognition of Landmarks. For even more reliable results, both Scanner and Landmark recognition may be used simultaneously.

Installation

For most applications, GotABot is installed on a host computer which can communicate with your robot. The recommended minimum requirements of the computer are:

- Pentium® III class, Intel® Celeron®, or AMD processor, or better - 1000 MHz or faster
- Windows XP or better (Windows XP must be updated with Net Framework)
- 256 MB RAM
- 50 MB of free Hard Disk Space

GotABot is a portable application and it may be unzipped or copied to any convenient directory.

3. Creating a Map

Maps are BMP files representing the environment the robot will operate in. They can be created by most graphics programs including Microsoft Paint.

- **Black** (Red=0, Green=0, Blue=0)areas represent walls, furniture and “out of bound” areas.
- **Grey** (Red=223, Green=223, Blue=223) represents areas which may be occupied or areas where the robot should not travel through unless it can find no other route.
- **White** (Red=255, Green=255, Blue=255) represents open space through which the robot is free to move.
- **Purple** (Red=128, Green=1, Blue=128) indicates areas which Scanners indicate as empty but which are solid objects or areas where the Robot should not go.

Scale

GotABot uses a map scale of 1 grid square = 1 square inch.

Creating Maps With Microsoft Paint

1) *Measure the actual area you want mapped – be accurate.*
2) *GotABot uses a map scale of 1 grid square = 1 square inch. As an example, a 10 x 12 foot space requires a map of 120 x 144 grid squares.*
3) *Start Paint (Start -> Run..->’mspaint’)*
4) *Set the Map Size (Press Ctrl+E then set the map Width and Height)*
5) *Set the Map Units (Press Ctrl+E then set the map units to 'Pixels')*
6) *Enlarge the map (Press Ctrl+PgUp)*
7) *Show a grid (Press CTRL+G)*
8) *Using the grid and the mouse position indicator in the bottom-left, draw the map.*

As an example, this is a map that I use:

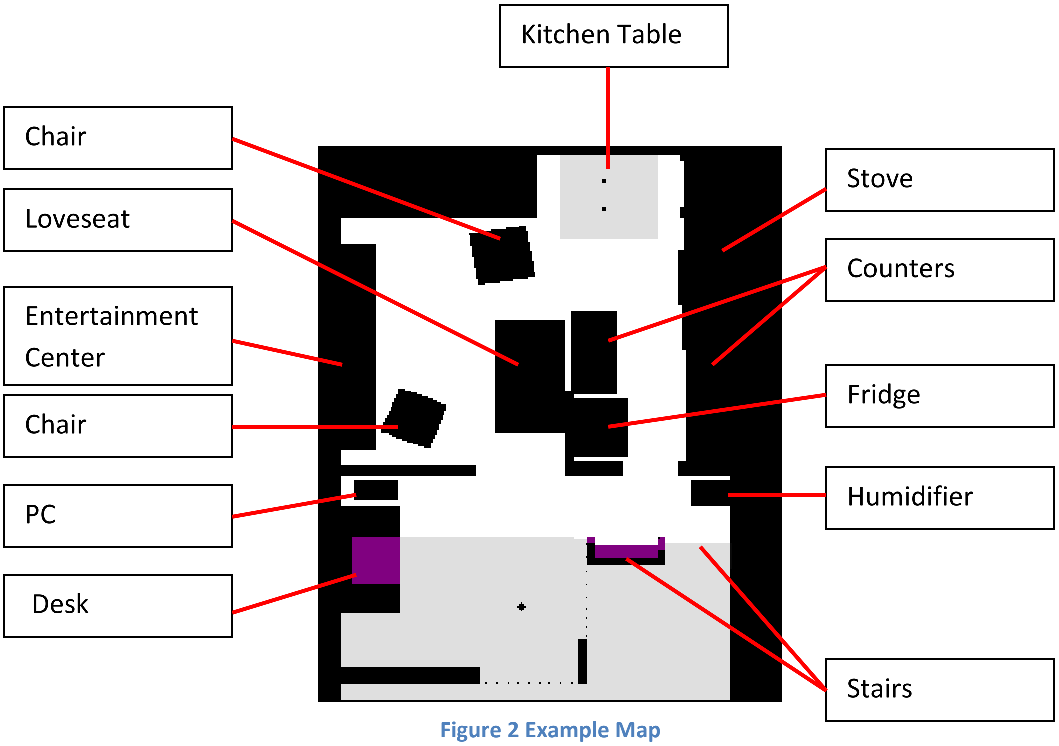


Figure 2 Example Map

…but shouldn't the robot make the map?

Well, it can (see the chapter on SLAM), but unfortunately, making a high accuracy map is a decidedly non trivial problem. Robot generated maps are especially problematic when the only measuring instruments are a cheap webcam, dodgy odometry and low cost range sensors. All in all, the best maps are drawn by old fashioned humans.

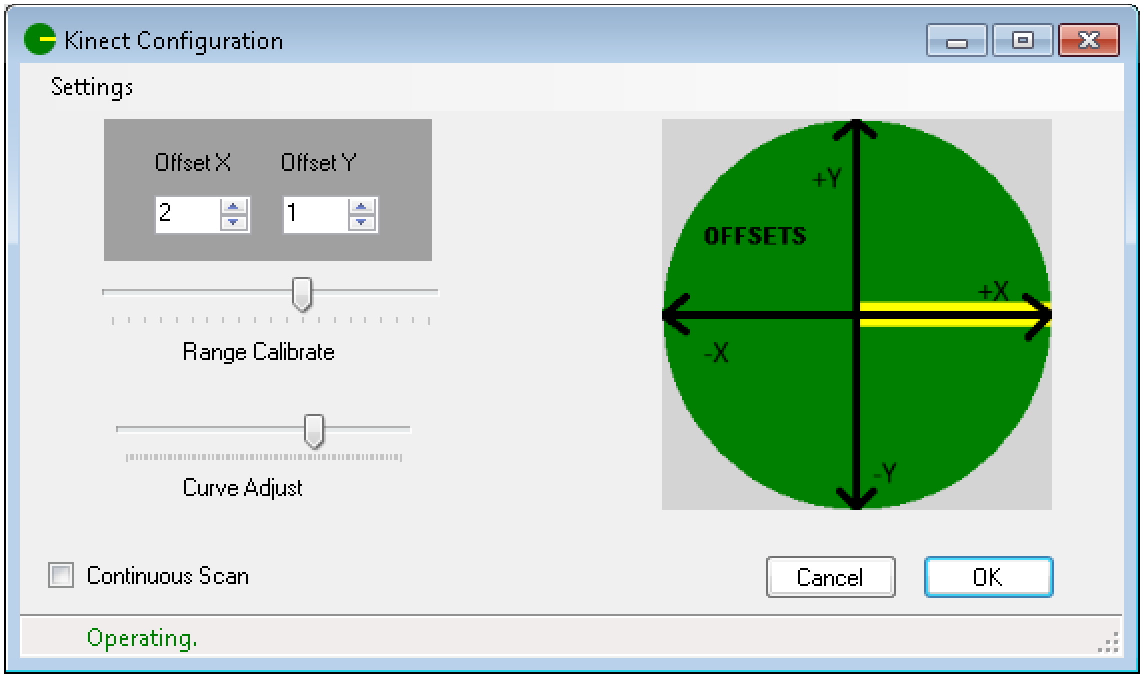
4. Scanner Configuration

GotABot provides interfaces to a Microsoft Kinect (first generation) or to a URG04 Laser Range Finder. Either will provide excellent localization and collision prevention. Localization using one of these Scanners is in many ways superior to using visual Land Marks as it is easier to configure, more accurate and localizes faster.

Configuration

Configure operation by opening the Scanner Configuration window (click *Configure->Scanner*).

The Offset fields represent the Scanner's position on the robot relative to the robot's center (typically centered on the drive wheels). Click the 'Continuous Scan' box and adjust the Calibration sliders to correct any errors in the range measurements.



Offset X: 2
Offset Y: 1

☐ Continuous Scan

Operating.

Figure 3 Scanner Configuration Window

TIP: Kinect Installation

The Kinect sensor requires an external power source beyond that applied through the USB port. Driver Version 1.8 is recommended. Install the SDK - not just the drivers.

5. Landmark Configuration

In addition to the Scanner interfaces, visual Landmark recognition may be used for determining the robot's position. Landmarks are user selected images that the robot can see through its webcam so as to recognize and measure range and direction to. Knowing the Land Mark's location and the estimated robot's position and orientation, GotABot uses simple geometry to determine where the robot is located.

For detection of Land Marks, GotABot provides interfaces to the Evolution Robotics RCC image recognition program (Available for download from the web).

RCC

This application is provided with the Evolution Robotics ER1 robot. Included in this application is a powerful image recognition utility suitable for identifying many types of images and useful for more than just ER1 control .

To configure this RCC application:

- i) click *Settings->Remote Control*
- ii) check '*Allow API control of this instance*'
- iii) set *Port* = 9000
- iv) *Password* = blank

(In the main GotABot window click *Configure-> Landmarks.* In the Landmark Configuration window click *Misc->Enable RCC.*)

Establishing a Land Mark is realized by:

1) Select the Landmark
2) Capture an image of it
3) Enter it in the GotABot database

Selection of Land Mark Images

Without a doubt, the selection of good Land Marks is the most crucial component to ensuring adequate localization. A few well chosen Land Marks will be the equal of dozens of randomly chosen images of a typical home environment.

A good Land Mark will be:

1) Unique
2) Easily recognizable by the robot's image processing algorithm
3) Useable over a wide range of distance and angles

In practice, 2D images such pictures, posters and playing cards are excellent choices. When first setting up, it would be well advised to temporarily attach a few of these to strategically located spots. Later, they can be replaced or augmented by a series of normal room images.

Tips For Good Land Marks

1) A good Land Mark should contain a number of high contrast features which the robot's algorithm can easily discern. Silhouette, or black and white, images are highly recommended!
2) Land Marks are most robust when they are in a single plane such as a picture, poster or playing card. Using such a Land Mark, the distance from robot to Land Mark can be estimated accurately over a wide range of distances and angles.
3) If Landmark images containing various features at various distances must be used, the valid region over which distance can be accurately measured will be relatively shallow. In such situations, it is best to make multiple Land Mark images, each at a slightly different distance.
4) Land Marks will yield more accurate range information when the orientation of the robot to Land Mark is close to that of the original image. If the Land Mark will be seen over a wide range of angles then create multiple Land Marks by capturing images at different angles.
5) Images which primarily consist of repeating features such as sets of drawers, slatted doors, patterned wallpaper or furniture, railings, etc, do not make good Land Marks.
6) Land Mark images which contain unwanted objects may be edited in Paint. Usually, the desired details may be easily isolated by overwriting everything else with grey rectangles.
7) Good locations for Land Marks include ends of corridors, doorways and any place where careful maneuvering is required to pass through tight spaces.
8) Ideally, the Land Mark should be at the same height as the camera.
9) Landmarks may not be recognized if they are placed against a 'busy' background. In such cases, a blank border around the land mark will improve recognition (i.e. if your landmark is, say, a

picture attached to a patterned background then insert a slightly larger blank sheet of paper between the picture and the background)

The Trouble With Land Marks

As a rule of thumb, there should be at least one Land Mark visible at any location the robot is likely to stop. With one Land Mark, the robot’s pose (i.e. – its position and orientation) can usually be calculated accurately, but sometimes, especially when the robot’s pose is not accurately known, errors will occur and the pose will be *shifted*. Usually, these errors will be corrected with time and further localizations, but sometimes the snowball accelerate downhill, the reported pose drifts ever further from reality and the robot runs into a wall.

The treatment for this error condition is simple – add more Land Marks. Make sure that wherever the robot is likely to look, there is a good Land Mark. If there are two Land Marks visible everywhere then so much the better. Errors introduced by one Land Mark will be corrected by the others.

Accuracy Counts!

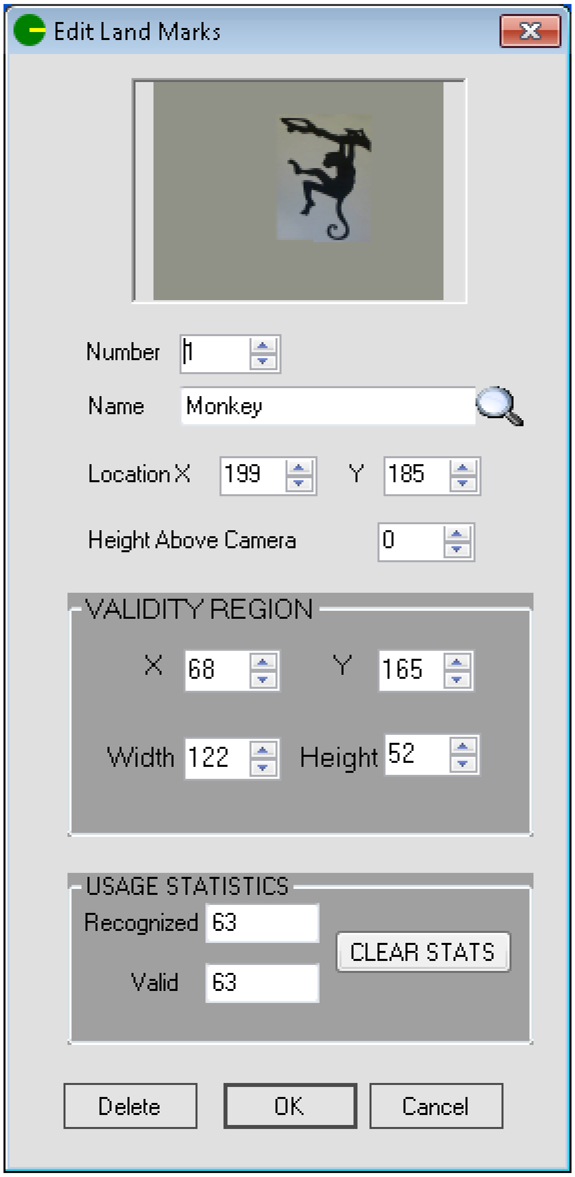
When establishing a Land Mark, make sure that the center of recognition is accurately transcribed in the Land Mark Edit form. It is probably necessary and certainly advisable to check where the image recognition software thinks the center is.

Having established where the actual center is, make sure that the range to this point is also correct.

The Edit Land Mark Window

Land Marks are entered through the Edit Land Mark window (In the main window click *Configure->Landmarks.* In the Landmarks Configuration window click *LandMarks*.)

. Remember to save your settings after making change s (*Click File->Save Settings*).



Edit Land Marks

Number 1

Name Monkey

Location X 199 Y 185

Height Above Camera 0

VALIDITY REGION

X 68 Y 165

Width 122 Height 52

USAGE STATISTICS

Recognized 63

Valid 63

Figure 5 Edit Land Mark Form

Land Mark Number

Each Land Mark has a unique number. As the number field is changed, the corresponding Land Mark and its Validity Region are plotted on the map.

Land Mark Name

The Land Mark Name field must contain the exact name reported by the image recognition program.

Search Land Mark Name

Pressing the Magnifying Glass icon beside the Name field allows a search of the Land Mark database by name.

Land Mark Location

This is the location of the center of the Land Mark. It may be entered by:

1) Click on either of the Location X or Y up / down entry box.
2) Click on the map at the location of the Land Mark.

Height Above Camera

This is an optional field, but it can help to calculate a more accurate range to a LandMark when the LandMark is not at camera level. This is typically useful when the camera is near ground level and the LandMark is, say, a picture hung at a much higher eye level.

Land Mark Validity Region

To lessen the chance of harm due to false Land Mark recognitions, a validity region is required for each Land Mark. This region restricts the area and robot orientation within which the recognition will be judged to be valid.

The validity area is a rectangle and is entered by:

1) Click on either of the Validity Region X or Y up / down entry box.
2) On the map, position the mouse at one of the Validity Region's corners, press the left button and draw a rectangle. Release the button.

Land Mark Usage Stats

These values are generated by the GotABot program and provide information on the usefulness of each Land Mark. Low Recognition or Validity numbers may be an indication that the Land Mark information is incorrect or that a better Land Mark is required.

6. Additional GotABot Configuration

A number of other parameters may also be set.

Robot Selection

Selection of a robot to control is achieved through this option (*Click Configure->Bot->Robot Select ->* and select your robot.

Pose

The robot's pose may be set with this option (*Click Configure-> Pose*). A robot's pose is a series of parameters which describes the robot's position on a map, the direction it is facing, and the uncertainties in each of these parameters.

Settings

GotABot's configuration setting may be saved or reloaded by selecting the "File" menu Settings options.

Robot Diameter

The robot's diameter may be set with this option (*Click Configure->Bot->Robot Diameter*). The Robot Diameter not only determines the smallest openings the Robot can move through, but it also sets the distance the Robot will try to stay away from walls and other obstructions. Setting the diameter too large will restrict the Robot from certain openings and/or may cause the Robot to perform many small "stop, turn and go" corrections while navigating narrow spaces. Setting the diameter too small will result in the Robot colliding with the walls. It's a tradeoff.

Camera Resolution

This setting will define the camera's resolution. (In the main window click *Configure->Landmarks.* In the Landmarks Configuration window click *Camera->Camera Resolution.*) Click on the resolution desired.

Camera FOV

This setting will define the camera's diagonal FOV or Field Of View (In the main window click *Configure-> Landmarks.* In the Landmark Configuration window click *Camera-> FOV.*)

Camera Offset

This setting will define how far the camera is from the *turning* center of the robot (a positive value indicates that the camera is closer to the front of the robot than the turning center) (In the main window click *Configure-> Landmarks.* In the Landmark Configuration window adjust the *Offset X* field.) Note – the turning center is usually centered between the two drive wheels.

Velocity

The robot’s linear velocity may be set with this option (*Click Configure->Bot->Velocity*)

Robot Acceleration

The robot’s linear acceleration may be set with this option (*Click Configure->Bot->Acceleration*)

Robot Angular Velocity

The robot’s angular velocity may be set with this option (*Click Configure->Bot-> Angular Velocity*)

TIP: Camera Selection / Configuration

Get the very best camera you can afford.

a) Choose a camera with a wide Field Of View. A minimum 60 degree FOV is recommended. More is better.

b) Choose a camera with good low light capability.

c) Choose the highest resolution the camera and PC speed will support. 1280 x 1024 works well. Even higher resolutions are better.

Robot Angular Acceleration

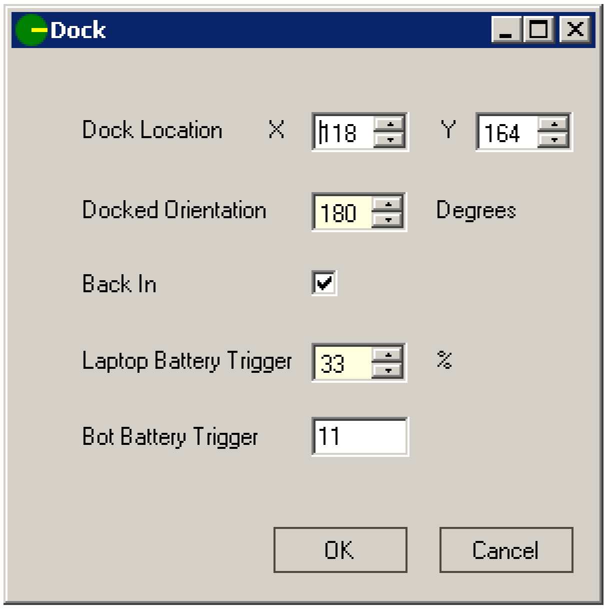
The robot’s angular acceleration may be set with this option (*Click Configure->Bot-> Angular Acceleration*)

Docking Configuration

Docking is the process which allows the Robot to (automatically) recharge its batteries.

To configure automatic docking click *Configure->Dock:*

1) Dock Location is the ***Robot*** location when docked.
2) Docked Orientation is the Robot's orientation when docked.
3) Back In should be checked if the Robot must back into the Dock.
4) The Laptop Battery and Robot Battery Trigger fields will trigger an automatic docking operation when either of the respective battery levels falls below the trigger level



Dock Location X 118 Y 164

Docked Orientation 180 Degrees

Back In ☑

Laptop Battery Trigger 33 %

Bot Battery Trigger 11

6 Docking Configuration Form

Localization Time

This setting may be used to control the length of time that GotABot will search for landmarks. Longer times will increase the number of times landmarks will be recognized (up to a maximum of 25 valid recognitions) thereby improving the localization accuracy. This setting may need to be increased if the image processing software is running on a slow PC or if other programs such as video servers are consuming the PC's time. (In the main window click *Configure-> Landmarks.* In the Landmark Configuration window click *Misc-> Visual Localization Time.)*

Mute

This setting prevents GotABot from speaking. (*Click Configure->Options->Mute*)

Password

Sets the Password used to verify API commands. (*Click Configure->Password*).

7. Operation

Software

By default, GotABot assumes that it is controlling an Evolution Robotics ER1 robot or a Heathkit Hero 1.

Operation Controls

Destination

The robot may be commanded to go to a destination by:

1) Click on the Tool Bar's Destination Icon (✗)
2) Click on the map at the desired position

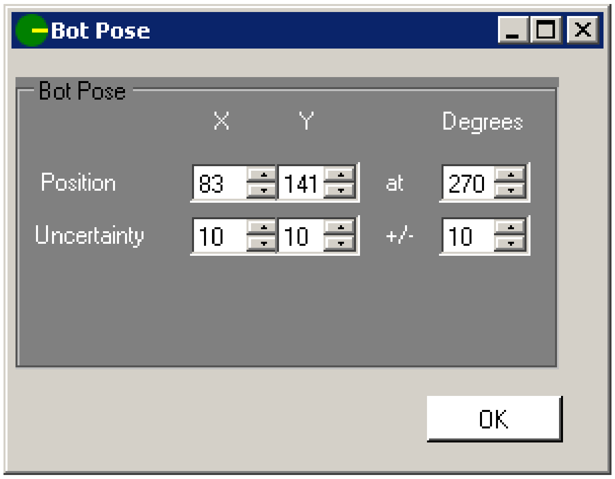
A yellow path (follow the yellow brick road) will be drawn on the map leading from the robot to the destination and the robot will be commanded to follow it.

Robot Position

The robot's position on the map may be changed by:

1) Click on the Tool Bar's Robot Icon (●)
2) Click on the map at the desired position
3) Click on the Tool Bar's Robot Orientation up/down controls (Orientation 0)
4) Adjust to the desired orientation

OR – Click *Configure-> Pose* and adjust as required.



Bot Pose

	X	Y		Degrees
Position	83	141	at	270
Uncertainty	10	10	+/-	10

Figure 7 Robot Pose Form

Dock

Docking is the process which allows the Robot to (automatically) recharge its batteries.
Docking Configuration is described in Docking Configuration. To manually initiate a docking operation click *Tools->Dock* .

Localize

Selecting this option (*Click Tools->Localize*) will force GotABot to use Land Marks and / or a Scanner to determine its position and orientation

Simulated Movement

Selecting this option (*Click Configure->Options->Simulated Movement*) will prevent GotABot from communicating with the robot. Instead, robot movement is simulated on the map.

Stop

Double clicking on this Tool Bar icon (STOP) will force the robot to stop.

Debugging

When something screws up (and it surly will) or acts in an unexpected fashion, it is helpful to review the robot's actions. A log file of GotABot's operations is provided for this purposes *(click File->Open Log File*). This file, *GotABot_Log.txt*, may be consulted to review movement, localization and other events.

Manual Controls

The robot may be controlled manually by using the keyboard’s Arrow keys (Forward, Back, Turn Left, and Turn Right). WARNING: Use caution with these controls. Rapid changes in direction could conceivably cause stress to the robot’s motor control electronics. Some robots are under engineered in this area.

8. Voice Commands

Voice commands are provided via the web service IFTTT and through Google Home or Alexa. If using Google Home a stop command could be issued by saying:

"*Hey google trigger stop*"

All of the following commands must be preceded by the appropriate wake phrase, either "*Hey Google*" or "*Alexa*". Commands which end with a number are further preceded by saying "*execute*". Except for "*go to*" commands, the other commands are preceded by saying "*trigger*". Note that all of the Command phrases may be modified in IFTTT.

COMMAND	**ACTION**
" *trigger stop* "	Stop all motion
" *execute forward 12*"	Move forward twelve inches
" *execute back 8*"	Move backwards eight inches
" *execute turn right 45*"	Turn 45 degrees right
" *execute turn left 30*"	Turn 30 degrees left
" *trigger status* "	Report status
" *trigger localize*"	Perform a localization operation
" *trigger recharge*"	Go to dock and recharge
" trigger look up"	Look up (requires Kinect or camera mounted on arm)
" trigger look down"	Look down (requires Kinect or camera mounted on arm)
" *trigger look at me*"	Look directly at nearest face
" *trigger my name is Bob*"	Tell the Face Recognizer your name
" *trigger remember my face*"	Train the Face Recognizer
" *trigger remember me*"	Train the Face Recognizer
" *trigger who am i*"	Say who is seen
" *trigger identify*"	Say who is seen
" trigger open"	open the gripper
" trigger close"	close the gripper
" trigger home"	return the arm to its home position
" *execute out 10*"	Move gripper forward 10 mm
" *execute in 22*"	Move gripper back 22 mm
" *execute up 115*"	Move gripper up 115 mm
" *execute down 37*"	Move gripper down 37 mm
" *go to kitchen*"	Go to the kitchen
" *execute pick up 3*"	Drive to and pick up object 3
" *trigger batch test*"	Execute batch file 'test.bat'
" *trigger guard mode*"	Detect movement near Bot
" *trigger photo*"	Take a photo
" *trigger quit*"	Exit and close GotABot
" *trigger shutdown*"	Exit and shutdown PC

Voice Setup

Device Required

Make sure you have a Google Home or Alexa voice controlled speaker. Each provides a superior voice recognition service.

Router Configuration

Configure your router to forward Ports 9100 and 9400 to the PC running GotABot.

DDNS

If your internet provider does not assign static IP addresses then sign up with one of the free DDNS services.

IFTTT Configuration

1) Goto IFTTT
2) Sign in or sign up for an account.
3) If you are using Google: (Alexa is similar)
    a) At the top right, click **Search**. Search for "Google Assistant."
    b) Click Google Assistant › **Connect**.
    c) Choose a Google Account to give IFTTT access to. Make sure to choose the account you used to set up Google Home or the Google Assistant on your phone. (Learn how to find the account you used to set up Google Home. If you're using the Google Assistant on a phone, touch and hold the Home button › More › **Account**. The account will be selected.)
4) How to create a Google Home Applet

Step 1: Create a phrase

Create one or more phrases you want to use to trigger an action from your Google Assistant.

1. Go to IFTTT.com.
2. At the top right, click your username › **New Applet**.
3. Click **this**.
4. Search for and click on"Google Assistant."
5. Choose the trigger. (see below)
6. Complete the trigger fields: . (see below)
7. Click **Create trigger**.

Step 2: Create an action

Create the action you want your Google Assistant to complete when you say your phrase.

1. Click **that**.
2. Search for Webhooks.
3. Choose "Make a web request" then fill in the required fields (see below).
4. Click **Create action**.
5. Review your Applet. To make changes, go to the top left, then click **Back**. If you're finished, click **Finish**.

5) Create the trigger Applet using the above method as a guide
   i) In the Google Assistant form
      (1) Choose trigger = "*Say a phrase with a text ingredient*"
      (2) Set "*What do you want to say?*" to "***trigger $***"
   ii) In the Webhooks form:
      (1) Choose "*Make a web request*"
      (2) Set the URL box to "***http://***_IPAddress_***:9100/exec?pw=***_Password_***&script={{TextField}}&end***"
         (i) replace *IPAddress* with your IP address or ddns server; and *Password* with your Password
      (3) Set Method to "***POST***"
      (4) Set Content Type to "***text/plain***"
   iii) Save your new applet.
6) Create the execute Applet. *Thi*s applet is triggered by a phrase followed by a number.
   i) In the Google Assistant form
      (1) Choose trigger = "*Say a simple phraset*"
      (2) Set "*What do you want to say?*" to "***execute $ #***"
   ii) In the Webhooks form:
      (1) Choose "*Make a web request*"
      (2) Set the URL box to " ***http://*** _IPAddress_***:9100/exec?pw=*** _Password_ ***&script={{TextField}}&xx= {{NumberField}}&end***"
         (i) replace *IPAddress* with your IP address or ddns server; and *Password* with your Password
      (3) Set Method to "***POST***"
      (4) Set Content Type to "***text/plain***"
   iii) Save your new applet.
7) Create the Go To Applet. This example instructs the robot to go to the kitchen.
   i) In the Google Assistant form
      (1) Choose trigger = "*Say a simple phraset*"
      (2) Set "*What do you want to say?*" to "***go to kitchen***"
   ii) In the Webhooks form:
      (1) Choose "*Make a web request*"
      (2) Set the URL box to "***http://***_IPAddress_***:9100/exec?pw=***_Password_***&script= set destination 150 90&end***"
         (i) replace *IPAddress* with your IP address or ddns server; and *Password* with your Password
         (ii) modify the destination coordinates of **150 90** to match your desired destination
      (3) Set Method to "***POST***"
      (4) Set Content Type to "***text/plain***"
   iii) You will need a separate applet for each location you want GotABot to go to.
   iv) Save your new applet.

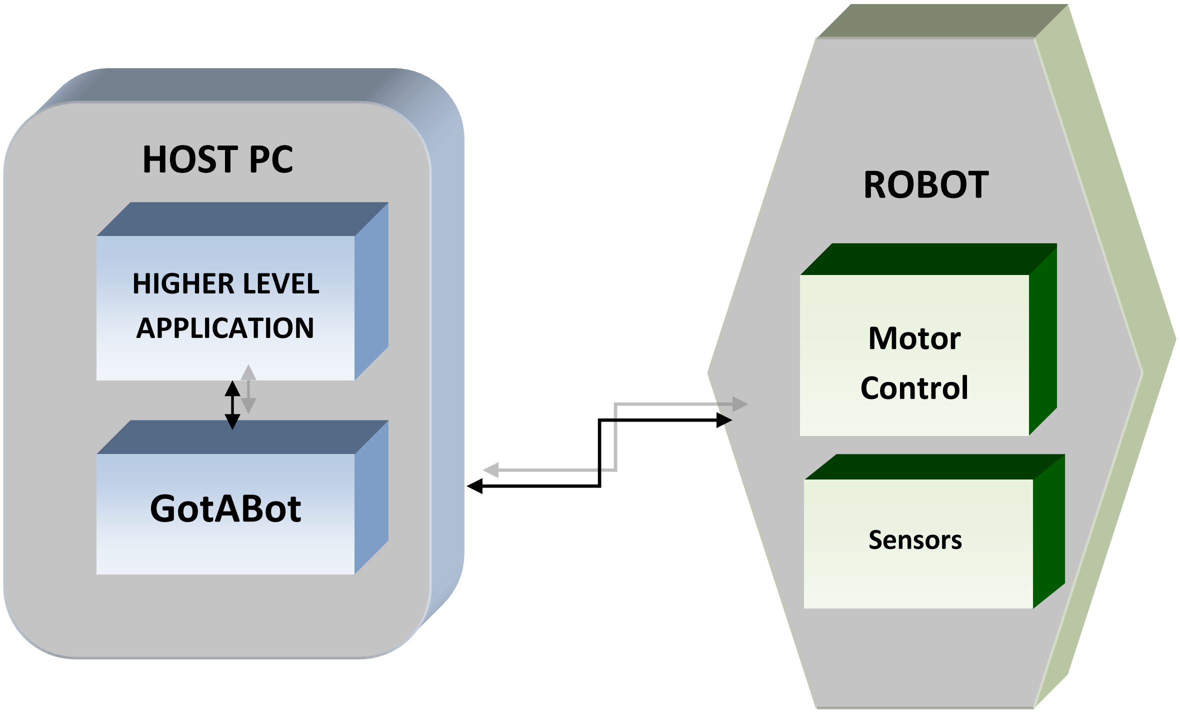
GotABot Configuration

i) Set the API password to match the Applet's (*Click Configure->API Password*)

9. API

Up to this point, we have discussed how GotABot may be controlled through the GUI (Graphical User Interface) or via voice commands. This chapter will explore an alternate control method, the API (Application Program Interface). The API is accessed by opening a TCP/IP socket to GotABot and using simple text commands to control GotABot and / or any robots connected to GotABot. This method of providing control is common and widely used by many robot control programs.

An API is especially useful when we have a higher level program or script which provides additional functions or behaviors for the robot. Theoretically, one could create a stack of API enabled programs, each relying and building on lower level functions to achieve new heights of robot behavior.



GotABot contains two identical APIs, either of which can be used by other programs or the user to pass commands or data to GotABot.

Accessing the API through Telnet

Accessing the API can be as simple as using a Telnet session. Although manual telnet control may have limited practical applications, it is very useful for debugging the commands and their sequences before implementing them in a script or program.

There are many different telnet programs but, as an example, here is how one could telnet to GotABot using the telnet program provided with Windows XP:

1) On the PC which will run the telnet program, click the **Start** button (lower left corner of your screen).
2) Click **Run** and type: *"telnet <ip address> <port>"*

Where the <ip address> is the ip address of the PC running GotABot
and <port> is 9100 for API #1 or 9400 for API #2
An example could be "*telnet 192.168.1.150 9100*"

3) Once connected, any of the API commands may be used.

TIP – Install Telnet Client for Windows 7, 8 or 10

As explained in http://technet.microsoft.com/en-us/library/cc771275(v=ws.10).aspx, the telnet client is not installed by default for Windows 7, 8 or 10.

TO INSTALL:

1. Open a command prompt window. Click **Start**, type **cmd** in the **Start Search** box, and then press **ENTER**.
2. Type the following command:

```
pkgmgr /iu:"TelnetClient"
```

3. If the **User Account Control** dialog box appears, confirm that the action it displays is what you want, and then click **Continue**.
4. When the command prompt appears again, the installation is complete

API Commands

GotABot supports the following API commands:

```
<PW> stop
<PW> pause
<PW> turn right xx
<PW> turn left xx
<PW> forward xx
<PW> back xx
<PW> set destination xx xx

<PW> set pose xx xx xx xx xx xx
<PW> set orientation xx xx
<PW> dock
<PW> localize
<PW> get pose
<PW> get status
<PW> get path xx xx

<PW> home
<PW> open
<PW> close
<PW> set gripper xx xx
<PW> pick up xx

<PW> look up
<PW> look down
<PW> look at me
<PW> remember my face
<PW> who am i

<PW> batchfile ssssss
<PW> exit application

<PW> help
```

(where xx is an integer, ssssss is a string and <PW> is the Password. All commands must be preceeded by the Password)

Turn Right Command

Instructs GotABot to turn right.

Example

COMMAND: *“turn right 45”*

RESPONSE: ”OK > “

ACTION: Instructs GotABot to turn right 45 degrees.

Turn Left Command

Instructs GotABot to turn left.

Example

COMMAND: *“turn left 30”*

RESPONSE: ”OK > “

ACTION: Instructs GotABot to turn left 30 degrees.

Forward Command

Instructs GotABot to move forward.

Example

COMMAND: “forward 17”

RESPONSE: ”OK > “

ACTION: Instructs GotABot to move forward 17 inches.

Back Command

Instructs GotABot to move back.

Example

COMMAND: *“back 6”*

RESPONSE: ”OK > “

Home Command

Instructs GotABot to move its arm to the home position.

Example

COMMAND: “home”

RESPONSE: ”OK > “

ACTION: Instructs GotABot to move its arm to the home position.

Open Command

Instructs GotABot to open its gripper.

Example

COMMAND: “open”

RESPONSE: ”OK > “

ACTION: Instructs GotABot to open its gripper.

Close Command

Instructs GotABot to close its gripper.

Example

COMMAND: “close”

RESPONSE: ”OK > “

ACTION: Instructs GotABot to close its gripper.

Pick Up Command

Instructs GotABot to drive to and pick up an object.

Example

COMMAND: “puck up 3”

RESPONSE: ”OK > “

ACTION: Instructs GotABot to pick up object 3.

Set Destination Command

This command is equivalent to the GUI’s Destination command.

Example

COMMAND: *“set destination 50 135”*

RESPONSE: ”OK > “

ACTION: Instructs GotABot to plan a path from the Robot to map position 50 135, and then to issue driving directions to the Robot.

Set Pose Command

This command allows the calling program to set the robot’s location, orientation and uncertainties. The command format is:

set pose <location x> <location y> <uncertainty x> <uncertainty y> <orientation> <orientation uncertainty>

Example

COMMAND: *“set pose 150 202 8 9 90 10”*

RESPONSE: ”OK > “

ACTION: This instructs GotABot that the robot is located at location 150, 202 with an 8 inch uncertainty in the X axis, 9 inch uncertainty in the Y axis, and with an orientation of 90 +/- 10 degrees.

Set Orientation Command

This command allows the calling program to set the robot’s orientation and uncertainty. The command format is:

set orientation <orientation> <orientation uncertainty>

NOTE: This command will only update the orientation when the robot is stopped and the orientation uncertainty is less than the existing uncertainty.

Example

COMMAND: *“set orientation 90 10”*

RESPONSE: ”OK > “

ACTION: This instructs GotABot that the robot’s orientation is 90 +/- 10 degrees.

Get Pose Command

This command will instruct GotABot to respond with the robot’s current pose in the format:

<location x> <location y> <uncertainty x> <uncertainty y> <orientation> <orientation uncertainty>

Example

COMMAND: *“get pose”*

RESPONSE: “POSE = *150 202 8 9 90 10*
OK > “

ACTION: This instructs GotABot to report the current Robot pose.

Get Status Command

This command will instruct GotABot to respond with the robot’s current driving status (*“docked” or “moving*” or “*stopped*”)

Example

COMMAND: “*get status*”

RESPONSE: ”stopped
OK > “

ACTION: This instructs GotABot to report the Driving status.

Dock Command

This command is equivalent to the GUI’s Dock command.

Example

COMMAND: “*dock*”

RESPONSE: OK > “

ACTION: This instructs GotABot to Dock with its charging station

Get Path Command

This command will instruct GotABot to plan a path from the robot to a destination, and then to respond with a series of locations indicating the steps required to reach that destination. The command format is: get path <location x> <location y> (where location x and location y are the desired destination)

Example

COMMAND: *"get path 150 55"*

RESPONSE: " Path =
171 188
178 181
178 177
180 175
180 160
181 159
181 157
182 156
182 125
176 119
176 77
154 55
150 55
OK >> "

ACTION: Plan a path to location 150, 55 and then report the steps to reach that destination.

Localize Command

This command is equivalent to the GUI’s Localize command.

Example

COMMAND: *“localize”*

RESPONSE: ” POSE = *150 202 8 9 90 10*
OK > “

ACTION: This instructs GotABot to perform a localize operation (look for Land Marks or perform a range scan , deduce the Robot’s position and orientation) and then to report the Robot’s pose.

Stop Command

This command is equivalent to the GUI’s Stop command. All active GotABot and robot activities should stop on receipt of this command.

Example

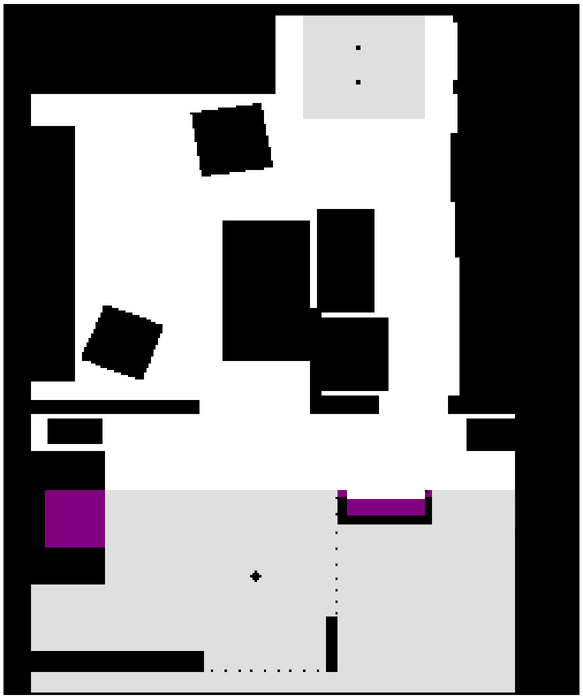
COMMAND: *“stop”*

RESPONSE: ” OK > “

ACTION: Stop all movement and localization.

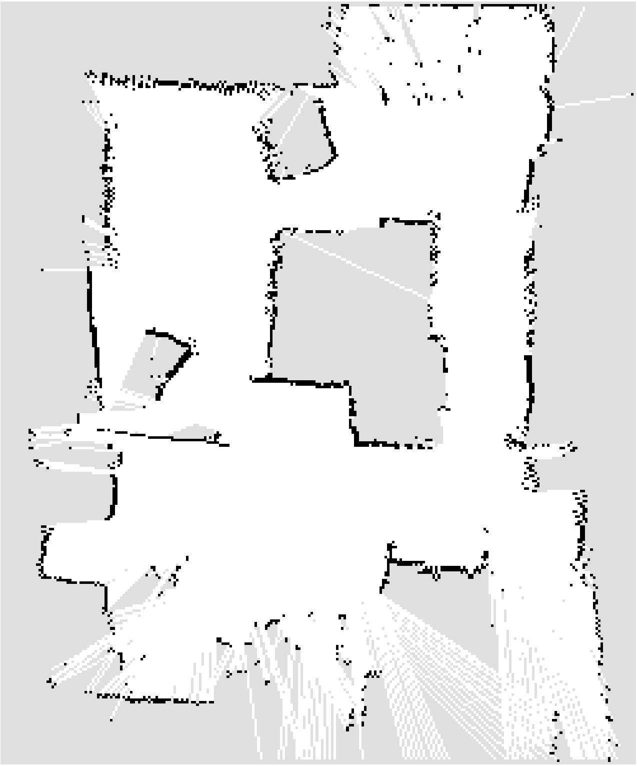
10. SLAM

SLAM, or Simultaneous Localization And Mapping, has long been one of the holy grails of the robot navigation world. After all, making a map by hand can be a fussy and aggravating exercise of measurement and transcription. Far better to let our clever robots handle these mundane details.

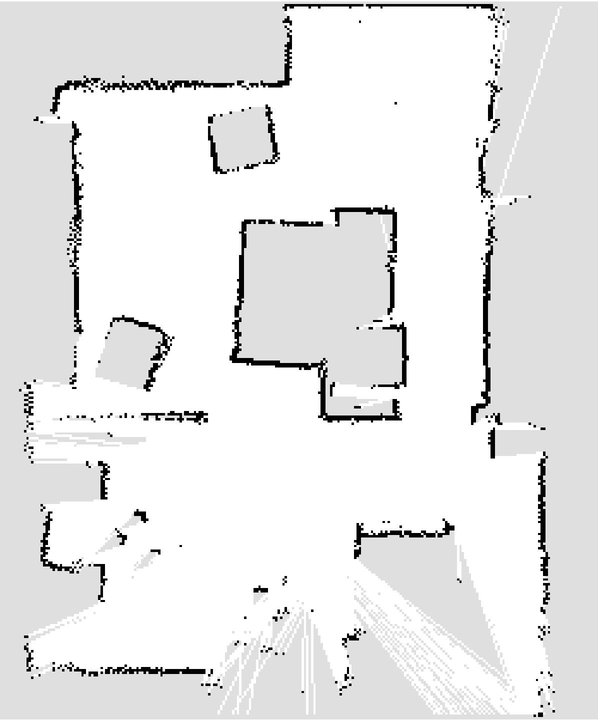


9 Hand Drawn Map

Over the years, many schemes utilizing various sophisticated algorithms and differing sensors have been developed and offered. Indeed, many academic careers have been built on this research.



8 Kinect Derived SLAM Map



10 URG04 Derived SLAM Map

At times, SLAM has been declared a ‘solved problem’ but new information on inadequacies of existing methods and new methods of addressing these inadequacies continue to be developed.

GotABot uses a variation of "Particle Filtering" for SLAM operations.

SLAM operation is started by clicking *Tools->SLAM*.

The Trouble With SLAM

In a word, the trouble with SLAM can be summed up as ‘accuracy'. Unless the Robot’s odometry or localization technique is exceptional and the ranging devices measure flawlessly, the resulting map will be less accurate, likely considerably less accurate, than a map generated by the user. For example, compare the user generated map with its equivalent Kinect derived SLAM map. Not as good, eh? Fortunately, even given the lower accuracy, the SLAM map is still adequate for most navigational purposes.

Kinect vs URG04 Map

As expected, the URG04 with its much wider FOV produces a map far superior to the Kinect's. Wall edges are more true with less noise and generally closer to their actual locations. More important than this is the ease of use in creating the map. The time required to map with the Kinect was several times longer and required judicious application of pirouettes and careful movements to keep the Particles from diverging and failing. By contrast, mapping with the URG04 was relatively easy, requiring only directing the Bot around the area to be mapped.

SLAM Requirements

The most obvious requirement for SLAM operation is, of course, to have some method of determining empty from occupied space. GotABot may employ a Kinect or a URG04 to achieve this.

SLAM Operation

To initiate and control the map making process, follow these steps:

STEP 1) Open the SLAM form by clicking *Tools->SLAM.*

STEP 2) Select a SLAM map when prompted. An unexplored SLAM map is a uniform grey (Red=223, Green=223, Blue=223). An example map is included in the installation package.

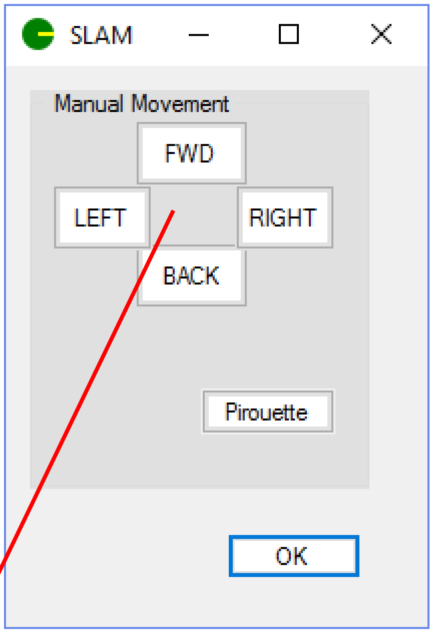


Figure 11 SLAM FORM

STEP 3) Use the Robot Pose form to set the map pose to match the actual robot's pose.

STEP 5) Use the *Manual Movement* controls to move and map.

STEP 6) Repeat step 5 until the map is adequately explored or the battery is drained.

STEP 7) Save the map. (Click *Configure->Map->Save Map).*

STEP 8) (Optional) The map may be enhanced or corrected by editing in a graphics editor such as MSPaint.

APPENDIX A: Evolution Robotics ER1

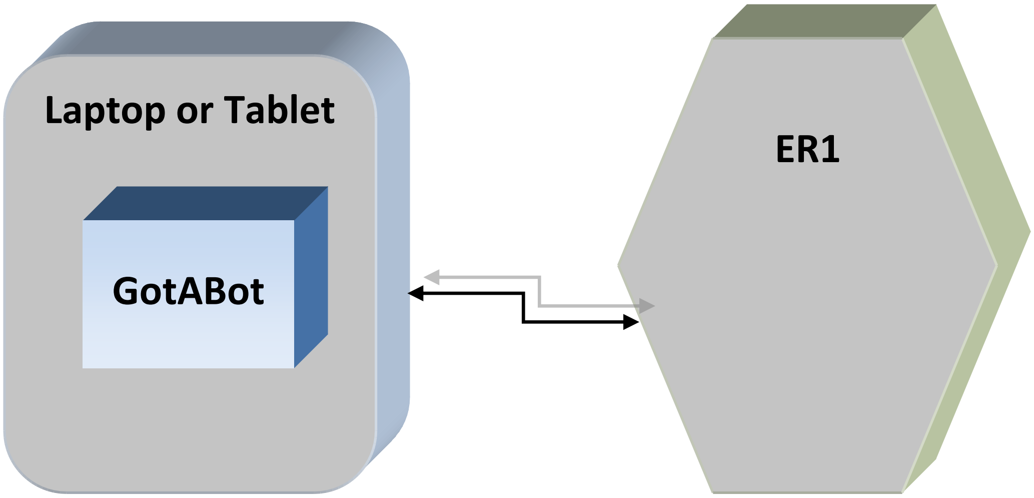
ER1 Driver Installation For Windows 10

The original hardware drivers do not work with any Microsoft Windows version post XP. To fix this, the GotABot download package contains new drivers which operate with more modern Windows.

NOTE. This driver installation uses modified FTDI VCP v2.12.12 drivers. They were modified by FTDI's FT_INF utility so as to work with the ER1's obsolete interface. This makes them unsigned drivers requiring a non standard installation.

To setup / use:

a. Enter the unsigned driver installation mode:
    i. Open the 'Run dialog' (*press Win+R*)
    ii. type `shutdown.exe /r /o /f /t 00`
    iii. After rebooting into the *'Options Menu*', click *'Troubleshoot'* and then *'Advanced Options*'.
    iv. Click *'startup settings*' and then '*restart*'.
    v. After rebooting, click option *7 'Disable driver signature enforcement*' and then reboot.
b. Navigate to the RCM Driver folder of the download package.
c. Right click on the file *ftdibus.inf* and click on *install*. Follow the prompts to install.
d. Right click on the file *ftdiport.inf* and click on *install*. Follow the prompts to install.
e. Connect the ER1. to your laptop.
f. Start GotABot
g. Select ER1.
h. Save settings and restart GotABot.



Optional ER1 Hardware

The ER1 has the capability of accessing both digital and analog ports through its Robot Control Module (RCM). GotABot exploits this capability by employing them to access various sensors and devices which extend and enhance its basic operation.

Use of these capabilities is optional and not required for basic operation. Robot's other than the ER1 may enjoy similar capabilities through use of the appropriate API command. ER1 bumper switches are enabled by clicking *Configure->Options->Bumper Switches*.

ER1 Bumper Switches

Gotabot supports three Bumper switches; left, right and rear. Behavior due to a Bumper Switch activation is the same as per the Bumper Switch API command.

Bumper Switch Wiring

RIGHT SWITCH

Switch Contact	*Signal Name*	*RCM J5-*
N.C.	+5V	1
N.O.	GND	25
Center	DIO	21

LEFT SWITCH

Switch Contact	*Signal Name*	*RCM J5-*
N.C.	+5V	1
N.O.	GND	25
Center	DI1	21

REAR SWITCH

Switch Contact	*Signal Name*	*RCM J5-*
N.C.	+5V	1
N.O.	GND	25
Center	DI2	22

TIP: Get Some Bumper Switches

Bumper Switches are such simple devices that their utility is frequently under appreciated. I would argue that given the inherent contrariness of a typical robot, bumper switches are well nigh indispensible.

APPENDIX B: Random Hints And Tips

Accessing a laptop while it is tethered to a robot is inconvenient. I use RealVNC (https://www.realvnc.com/) to login from my desktop. Much easier.

If GotABot is installed in a Windows XP system then make sure that the system has been updated with Net Framework.

If the robot 'fishtails' when coming to a stop, increase the 'Power Stopped' (press *Configure->Bot->Power Stopped*).

A camera or a Kinect will see more close in details if it is mounted near the rear of the robot.

For faster localization, use the smallest map which still shows the area which the robot can observe and operates within.

Recommended Speeds for the ER1:

Acceleration:	60 cm/s^2
Velocity:	20 cm/s
Angular Acceleration:	90 Degrees/s^2
Angular Velocity:	40 Degrees/s

Example Power Settings for the ER1:

Power Moving:	65
Power Stopped:	6

APPENDIX C: Heathkit Hero-1

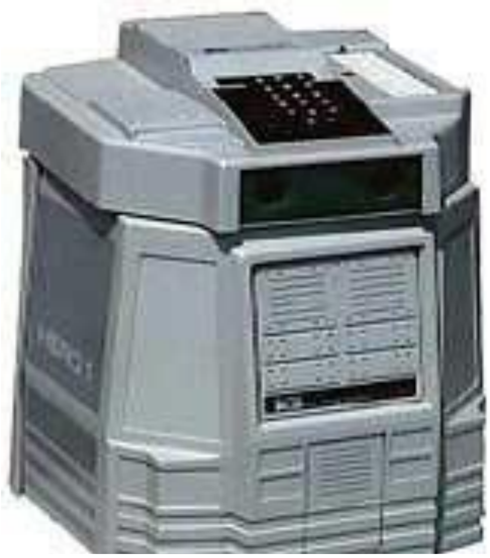


Figure 12 Heathkit Hero-1

The Heathkit Hero-1 is one of the all time great hobbyist robots. First produced in 1982, it has all the features required by GotABot; mobility, odometry, range sensing and an interface. Unfortunately, its early technology results in limited sensor range and discrimination, poor steering accuracy, and glacially slow processing. Still, extending this iconic platform's capabilities with a modern laptop or tablet is educational and a hoot to boot.

Hero-1 / GotABot Configuration

Follow these steps:

1) The Hero must have a Serial or USB interface as well as a the requisite 1.U or 1.3 ROM with 4 MHz crystal. Configure the interface for 9600 baud.
2) Create a map.
3) Connect the laptop or tablet to the Hero. (See example).
4) Turn on the laptop and Hero.
5) Initialize Hero by pressing *RESET* then *3* then *1* on the keypad
6) Start GotABot.
7) Click *Configure->Bot->Bot Com Port* and select the port connected to the Hero.
8) Click *Configure->Bot->Bot Select* and select *Heathkit Hero.*
9 Click *File->Save Settings* and then restart GotAbot.
10*)* Follow the onscreen instructions to download an assembly program to the Hero.
11) Test your setup by Localizing and Manual movement control.
12) Job Done!

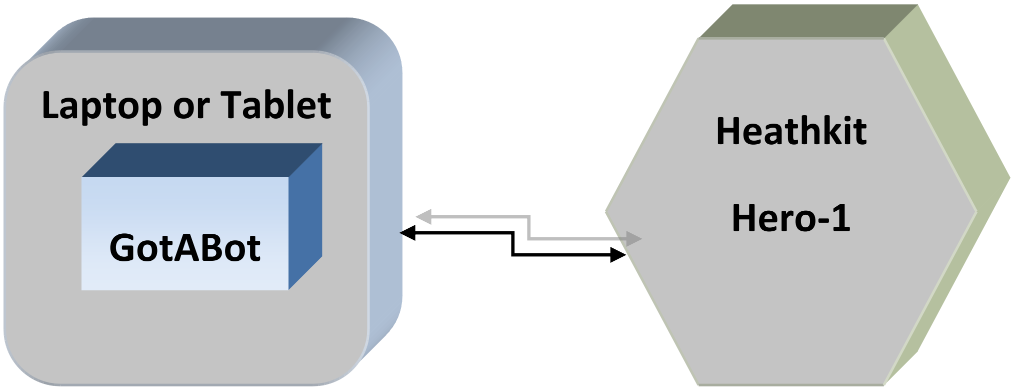


Figure 13 Hero-1 / GotABot Configuration

Hero To Tablet Connection Example

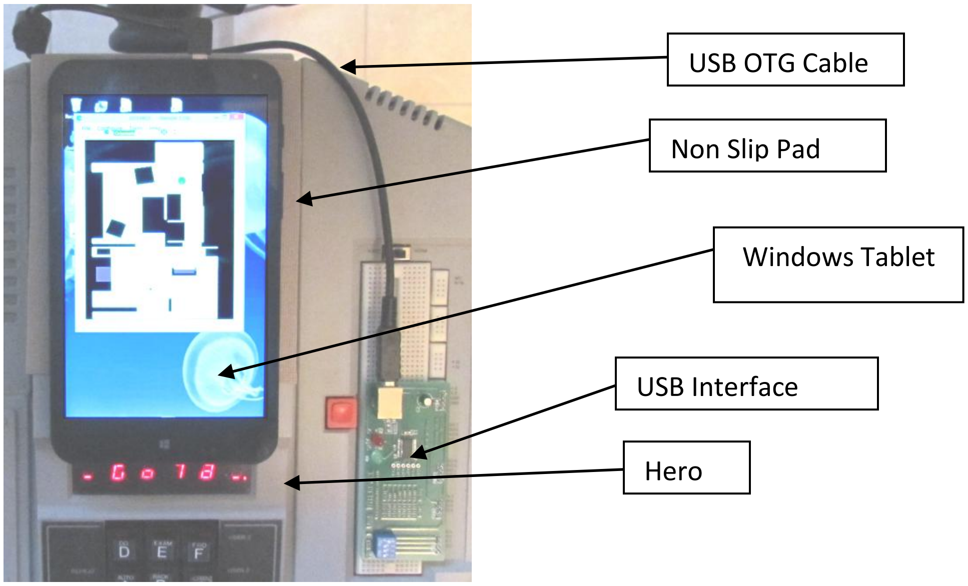


Figure 14 Hero To GotABot USB Connection

Laptop Remote

Operating the laptop while it is being carried hither and yon is challenging and inconvenient. I use VNC to log into the laptop from my desktop.

Hero-1 Hardware Compensation

If your Hero's hardware is not perfectly aligned then there are some settings which might compensate.

Hero-1 Steering Compensation

If your Hero veers left or right instead of going dead ahead, then click *Configure->Bot->Steering Bias Correction* and enter an adjustment.

Hero-1 Head Offset Compensation

If your Hero's head does not point straight ahead then click *Configure->Scanner* and adjust the *Scanner Offset in Degrees* field to compensate.

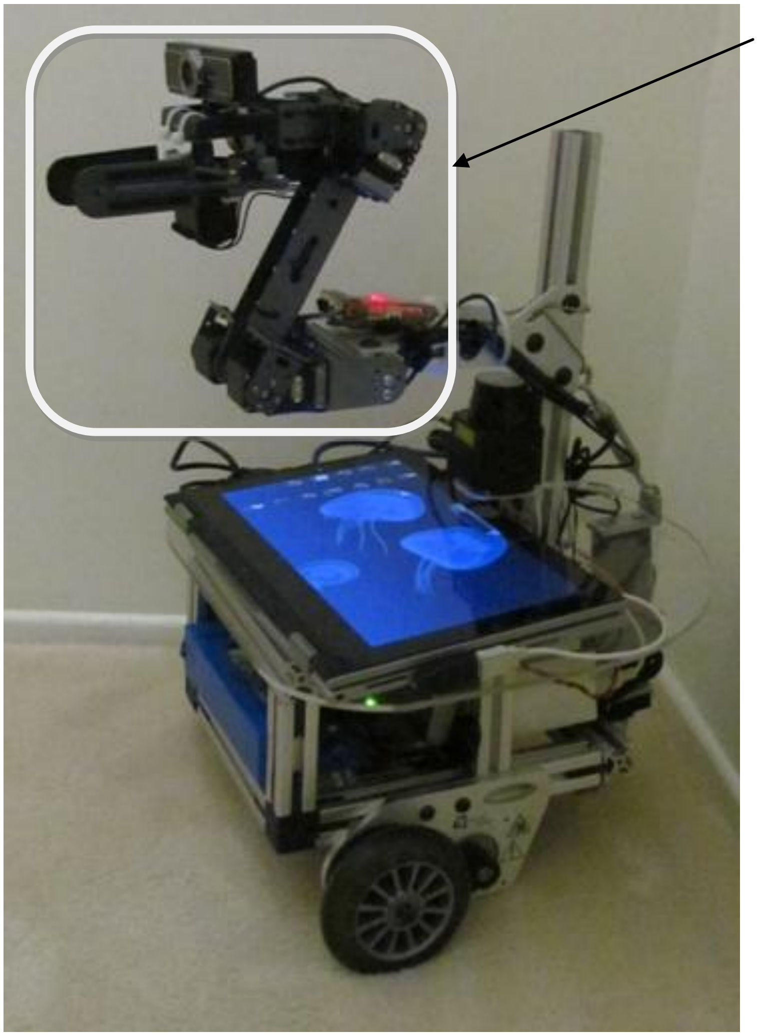
Hero-1 Distance Compensation

If your Hero does not accurately travel distances commanded then click *Configure->Bot->Distance Compensation* and enter the polynomial coefficients which describe its actual response.

TIP: Keep it simple

As discussed, the Hero-1 has limited sensing and movement accuracy. Successful operation will likely require an environment with extra wide pathways and uncluttered walls which return good echoes for the ultrasonic ranger.

APPENDIX E: Arm



GotABot's arm is a CrustCrawler AX-12AUSB with the rotating base removed and mounted as shown. A Robotis USB2Dynamixel is used to interface between the arm and laptop.

NOTE: The Wrist Rotate servo is mounted upside down (vs the original build instructions)

Joint	**Servo Number**	**Direction**
Shoulder	2	Reverse
Shoulder	3	Forward
Elbow	4	Forward
Elbow	5	Reverse
Wrist Rotate	6	Forward
Gripper	7	Forward

APPENDIX F: GBot

So, your RCM is not working and your ER1 is useless. Perhaps old age, random chance or a poorly conceived experiment has broken it. Happens. GBot is a replacement which works with the ER1 hardware (but not the original RCC software. No matter - it works with GotABot.)

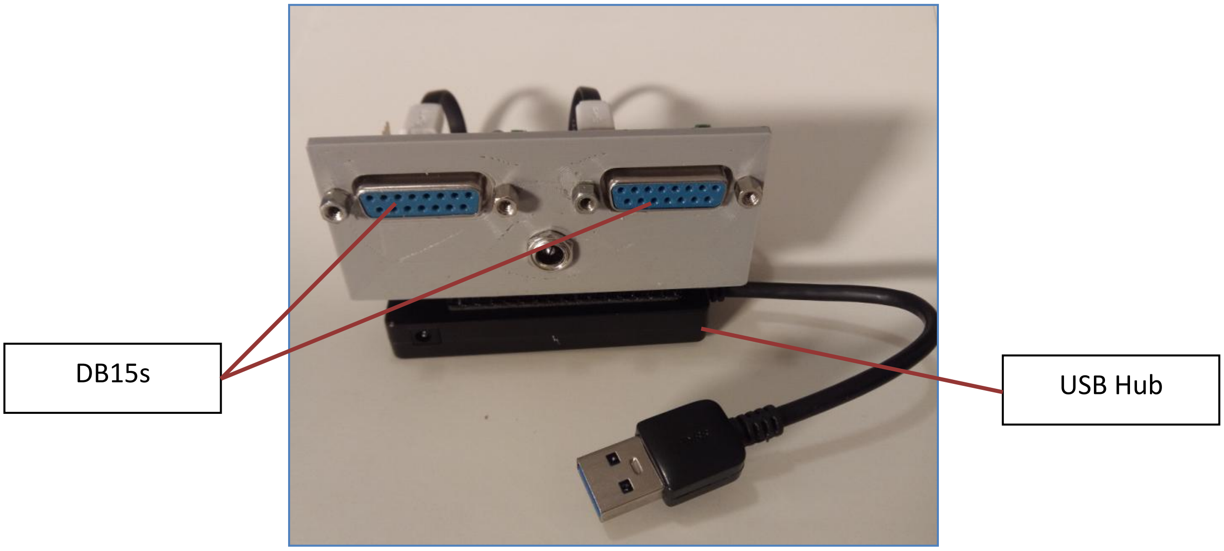


Figure 15 GBot Controller Front

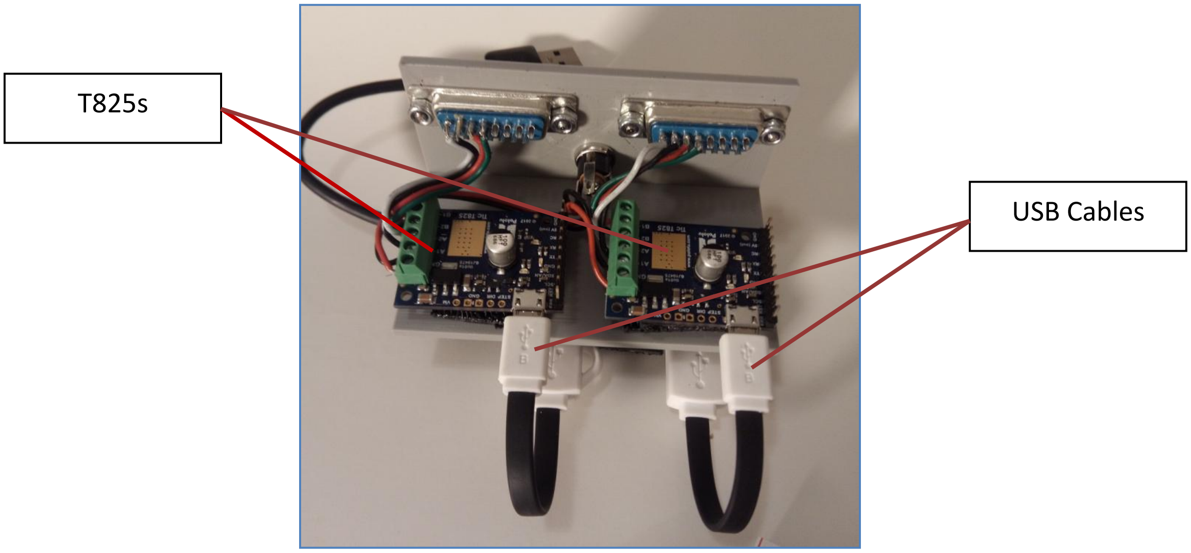


Figure 16 GBot Controller Rear

Qty	PARTS LIST	US$
2	Pololu T825 Stepper Controller	60.00
2	DB15 Female Panel Mount	4.00
2	USB cables USB A to micro-B	7.00
1	Power Jack	2.00
1	USB Hub	10.00
1	Enclosure (3D Printed or original RCM Enclosure)	
TOTAL		**83.00**

Table i GBot Parts List

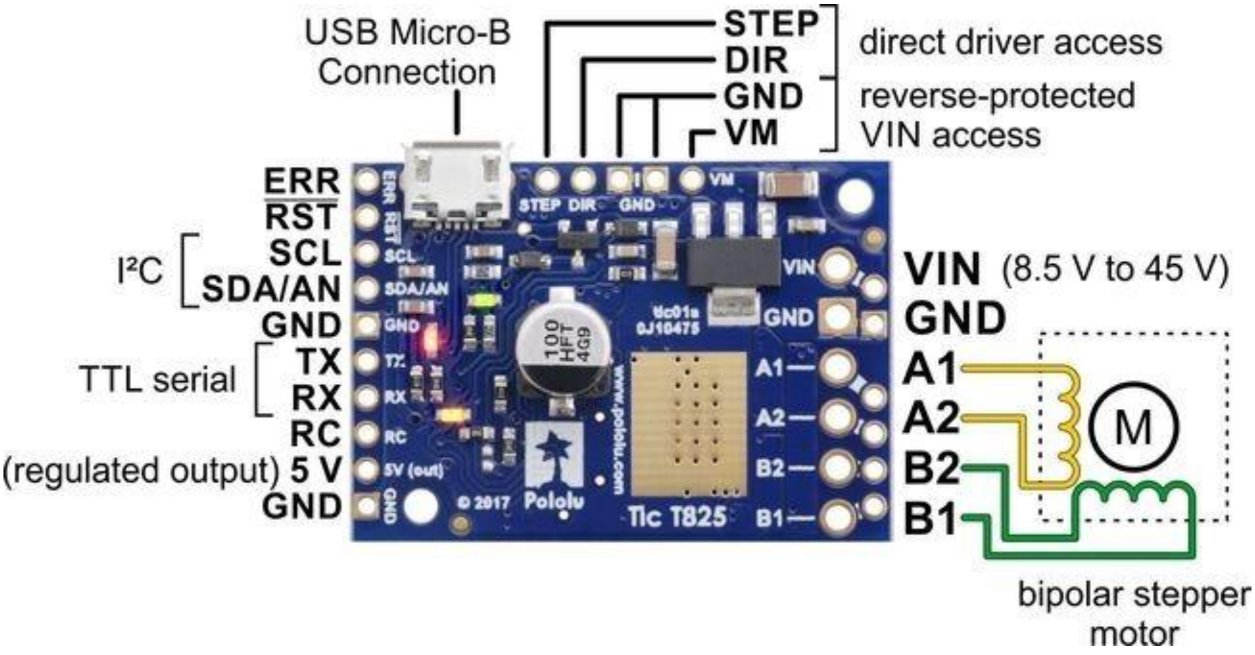


Figure 17 Pololu T825

GBot Wiring Table

Pololu T825	
A1	DB15 - pin 2
A2	DB15 - pin 1
B2	DB15 - pin 4
B1	DB15 - pin 3
Vin	Power Jack - +12V
GND	Power Jack - GND

Table ii GBot Controller Wiring

Setup

1) Download the T825 Software and Drivers from Pololu.
2) Install and select the Add To Path option
3) Connect the T825s to power and the PC.
4) Open the ticgui.exe utility. For each of the T825s (select via the 'Connected to' box):
    a) Select the 'Input and motor settings' tab
    b) Unselect the 'Enable command timeout' box
    c) Set the Baud rate to 115385
    d) Click 'Apply settings'

GLOSSARY

API	Application Program Interface.
Bot	Short for Robot.
ER1	Robot manufactured by Evolution Robotics.
FOV	Field Of View. The diagonal angle, in degrees, captured by the camera.
IFTTT	If This Then That (Web application for voice control via Google Home or Alexa)
Land Mark	User selected images that the robot can recognize and measure range and direction to.
Localization	Determination of a Robot’s pose through the use of Land Mark recognition or other sensory input.
Pose	The robot’s position in space, quantified in the x, y and theta dimensions.
RCC	Robot Control Center. Software for image recognition.
RCM	Robot Control Module. Hardware component of the ER1.
Recognizer	A program which uses the robot's camera and performs object recognition on the image . This program must be able to determine the range to the object and relative angle. The Recognizer is integral to the Landmark recognition process.
SLAM	Simultaneous Localization And Mapping
USB OTG	On The Go. Enables devices such as some Tablets to act as USB hosts.
VGA	The VGA or Video Graphics Array standard specifies a resolution of 640 by 480 pixels.
YMMV	Your Mileage Might Vary.